Rescheduling and Checkpointing as Strategies to Run Synchronous Parallel Programs on P2P Desktop Grids

Rodrigo da Rosa Righi, Alexandre Veith, Vinicius Facco Rodrigues, Gustavo Rostirolla, Cristiano André da Costa, Kleinner Farias
Applied Computing Graduate Program - Unisinos - Brazil
{rrrighi,veith,vfrodrigues,rostirolla,cac,kleinnerfarias}@unisinos.br

Antonio Marcos Alberti
Instituto Nacional de Telecomunicações
INATEL - Brazil
alberti@inatel.br

ABSTRACT

Today, BSP (Bulk-Synchronous Parallel) represents one of the most often used models for writing tightly-coupled parallel programs. As resource substrates, commonly clusters and eventually computational grids are used to run BSP applications. In this context, here we investigate the use of collaborative computing and idle resources to execute this kind of demand, so we are proposing a model named BSPonP2P to answer the following question: How can we develop an efficient and viable model to run BSP applications on P2P Desktop Grids? We answer it by providing both process rescheduling and checkpointing to deal with dynamism at application and infrastructure levels and resource heterogeneity. The results concern a prototype that ran over a subset of the Grid5000, showing encouraging results on using collaboration and volatile resources for HPC.

Categories and Subject Descriptors

C.1.4 [**Parallel Architectures**]: Distributed architectures; D.4.1 [**Process Management**]: Scheduling; H.3.4 [**Systems and Software**]: Distributed systems

General Terms

Performance, Management

Keywords

Bulk-Synchronous Parallel, P2P, Process Rescheduling, Checkpointing, Performance

1. INTRODUCTION

Bulk Synchronous Parallel (BSP) represents a common model for writing successful parallel programs that exhibit phase-based computational behaviors [5]. As deployment machines, this programming model has been used on clusters and computational grids [6]. Particularly, this kind of structure is known by normally presenting a centralized or hierarchical architecture, high-speed networks linked to the Internet and nodes that slowly change their participation behavior along the time [6]. In addition, for using one of the aforementioned parallel machines, the user must either buy the computational and network infrastructures or present a previous contract/agreement with the institution that host it. Concerning this landscape, we started the study of low cost and collaborative environments to take profit of end nodes around the Internet effortlessly. This effort culminated in an architecture proposed by Zhao, Liu and Li named P2P Desktop Grids [16] (PDG). Although joining the power of idle resources, a high level of dynamism with the sudden leaving of users, worldwide-scale and Internet-based connections are challenges when associating this environment with the purpose of HPC. In this way, our work presents the following problem statement: *How can we explore collaborative computing on PDG to run BSP applications efficiently?*

Aiming at answering the posed question, we are proposing **BSPonP2P** - a model that encompasses an infrastructure, overlay network, scheduling algorithms and runtime management to run BSP applications on PDG. BSPonP2P addresses collaborative computing at middleware level, where programmers do not need to change their applications in order to execute them in the P2P setting. Consequently, the proposed model acts as a middleware to run round-based parallel applications with spatial decoupling in an easier and costless way. Moreover, we also present its evaluation with a BSP application in a real infrastructure of multiple clusters.

The remainder of this article will first introduce the related work in Section 2. Section 3 describes BSPonP2P in detail, demonstrating its rationales and contributions. Evaluation methodology and the discussion of the results are presented in Sections 4 and 5, respectively. Finally, Section 6 emphasizes the scientific contribution of the work and notes several challenges that we can address in the future.

2. RELATED WORK

This section briefly presents initiatives to run applications on collaborative environments. Focusing to support Bag-of-Tasks (BoT) applications, the authors in [1] present a generic content-based publish/subscribe system called DPS. Moreover, Leite et al. [8] propose a load balancing architecture

SAC'15 April 13-17, 2015, Salamanca, Spain.

http://dx.doi.org/10.1145/2695664.2695979.

using a P2P-like structure for desktop grids.

Besides BoT, the master-slave approach is addressed in [4, 12, 13, 15]. Balasubramaniam et al. [2] and Byung et al. [14] presented aproaches targeting Desktop Grids, and Godfrey et al. [4], Shudo [13], Senís et al. [12] and Wu and Tian [15] present aproaches targeting PDG.

Concerning BSP parallel applications, both Mizan [7] and Camargo et al. [3] are representative for heterogeneous and dynamic environments. Mizan is a dynamic load balancing that captures data from computation and communication metrics. Camargo enable the use of not only idle processor cycles, but also unused disk space of shared machines, and a checkpointing-based mechanism.

Table 1 presents a summary of the aforementioned systems and algorithms. As shown, the initiatives approach different models for collaborative environments and assorted scheduling strategies. We can note that few works are focusing on metrics different of computation, as well as on failure control. In this regard, we observe a research opportunity to work with tightly-coupled applications, such as BSP, on collaborative environments, offering pertinent strategies to cover BSP features on highly dynamic and heterogeneous substrates.

3. BSPONP2P: PROPOSAL TO RUN BSP PROGRAMS ON P2P DESKTOP GRIDS

BSPonP2P architecture was developed taking in mind both structured and unstructured P2P networks as shown in Figure 1. Firstly, we are working with a structured ring-based network following the Chord P2P protocol [9]. This kind of network is used to connect nodes, named as Managers. Each Manager is responsible for a specific cluster, where the cluster here means a parallel machine, a local network, a mobile device, or a single computer. A cluster is organized in an unstructured manner, since it offers better flexibility and dynamism with heterogeneous and unstable resources. The nodes inside a cluster are named End Nodes, or only Nodes, and they are responsible to execute the BSP applications.

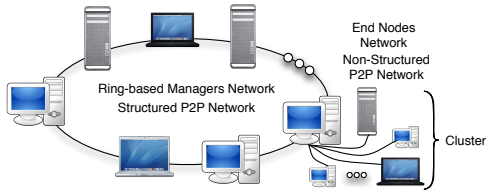

Figure 1: Computational Overlay Network with two communication levels: (i) among the Managers; (ii) between a Manager and an End Node.

Each resource can act as a Manager or End Node. We created a Computational Overlay Network (CON) to manage message routing, scheduling, as well as the entrance and the leaving of a resource in the infrastructure. Each cluster has at maximum n End Nodes. Thus, the first resource will act as a Manager and the others up to n will serve as End Nodes to the composed cluster. Aiming at getting End Node data periodically, a Manager sends query requests at intervals of t_m seconds (defined by the Manager). If the Manager does not receive a response from an End Node two consecutive times, the node is disconnected from the CON.

CON automatically reorganizes the network when a node suffers a crash or intentionally leaves the collaborative infrastructure. This node can be either a Manager or an End Node. In this case of a Manager, the oldest End Node in the cluster is promoted to be the Manager. Another possibility consists of an outgoing of an End Node that had its computation abruptly interrupted. In this context, the Manager has partial data about the execution and can select other peer in accordance with the scheduling function. Since BSPonP2P works with process checkpoiting it is possible to continue application execution from the last saved point.

BSPonP2P also uses the process migration strategy to treat the heterogeneity of resources in an efficient way to properly run a phase-based application. This strategy is provided at middleware level, not imposing modification in the user's application. The evaluation of the first level will decide which cluster will execute a particular process. For that, we are using a decision function denoted PM (Potential of Migration) proposed by Righi et. al. [10] in the MigBSP approach. PM is computed through Equation 1, which receives as inputs i and j, a process and a cluster, respectively. In this context, $Comp$, $Comm$ and Mem denote computation, communication and memory metrics. The larger the PM value, the most profitable is the target cluster j on receiving a process i. Different from MigBSP, BSPonP2P uses PDG, which implies in a modification of the computation metric in accordance with Equation 2. $T(i)$ and $Set(j)$ are inherited from MigBSP [10], and denote the computational time of process i in the last superstep and the relative performance of the cluster j, respectively. BSPonP2P adds $\overline{X}_{Resource}$ and $\overline{X}_{User}$ in order to evaluate the resource utilization in a cluster and the resource utilization by the user.

$$PM(i,j) = Comp(i,j) + Comm(i,j) - Mem(i,j) \quad (1)$$

$$Comp(i,j) = (\frac{\overline{X}_{Resourc(j)} + \overline{X}_{User(j)}}{2}).T(i).Set(j) \quad (2)$$

The evaluation in the second communication level is used to define which End Node in a cluster will run a specific process. The definition of the executor node is made based on the availability of the equipment. At this point, a simple assessment is made, where samples of at least three ratings and a maximum of ten reviews of availability (amount of computational resource available) are used. The samples are based on past records received by the Manager.

As runtime strategies, BSPonP2P offers process rescheduling and checkpointing. Both take place after ending a particular superstep, because of this point refers to a consistent global state of the distributed system. The idea is to offer a runtime management that aims at reducing the load imbalance among the processes, so decreasing the execution time of each superstep. Rescheduling tests are done not at each superstep, but the superstep index is defined on-the-fly in accordance with the MigBSP parameter called α.

4. EVALUATION METHODOLOGY

The evaluation was performed using SimGrid[1], a deterministic scientific instrument to study the behavior of scheduling algorithms in heterogeneous platforms. We applied sim-

[1]http://simgrid.gforge.inria.fr

Table 1: Comparison among initiatives to run parallel applications on collaborative environments.

Initiatives	Target system	Model application	Migration	Data Replication	Load balancing	Monitoring
Anceaume et al. [1]	PDG	Bags of Task	-	-	Computation	Computation
Balasubramaniam et al. [2]	Desktop Grids	Master/Slave	-	-	Computation	Computation
Camargo et al. [3]	PDG	BSP	yes	yes	Computation	Computation
Godfrey et al. [4]	PDG	Master/Slave	yes	yes	Computation	Computation
Khayyat et al. [7]	Desktop Grids	BSP	yes	no	Computation	Computation
Leite et al. [8]	PDG	Bags of Task	yes	yes	Work Stealing	Computation
Sentís et al. [12]	PDG	Master/Slave	yes	no	Computation	Computation and Memory
Shudo et al. [13]	PDG	Master/Slave	yes	no	Computation	Computation
Byung et al. [14]	Desktop Grids	Master/Slave	yes	no	Computation	Computation
Wu et al. [15]	PDG	Master/Slave	yes	yes	Computation	Computation

ulation in the three scenarios using the Simgrid's MSG module, using as platform the first 15 nodes of the following Grid5000 clusters[2]: chimint, chicon, paradent, grephene, gdx, capricorne, adonis, borderplage, pastel and suno.

Scenario i represents the simple execution of a BSP application, disabling any service or scheduling functionality. Scenario ii adds the scheduling calculus in the first and second levels of the CON. Finally scenario iii enables process checkpointing and rescheduling. The objective is to show the overload imposed by BPSonP2P (comparing scenarios i and ii), and the gain or loss of time when migration is enabled (comparing scenarios i and iii). We also performed a recovery validation: in this case, we evaluated the time to resume the execution with checkpoint and compared with the time without this service.

We implemented a BSP application for computational fluid dynamics based on the principle of the Lattice Boltzmann Method (LBM) [11]. Tests conducted on each scenario suffered the variation of three parameters: (i) α, starting with 4, 8 and 16 (same values used by [10]); (ii) Amount of supersteps whose values tested were 10, 50, 100, 500, 1000 and 2000; (iii) Amount of processes, assuming the values 11, 26, 51 and 89, randomly chosen to represent the environment that is found on PDG.

5. DISCUSSING THE RESULTS

Aiming at analyzing the changes in the execution time according to the variation of the parameters presented in the previous section, the results were organized in Figure 2 according to the number of processes showing the percentage change in scenarios ii and iii in comparison to scenario i.

Analyzing the results, we can observe that, when α is equal to 4 and the amount of supersteps is equal or bigger than 1000 scenario iii is always better than scenario ii. In cases with 51 and 89 processes and 1000 supersteps a gain of 8% and 16% respectively can be obtained above scenario i. With α equal to 8, scenario iii is better than scenario ii in almost every test losing only when the amount of supersteps is 500 with 89 processes by 0.56%. In these cases, the ROI (Return of Investments) of migrations reaches almost 17% of gain above scenario i with 89 processes and 1000 supersteps. When scenario i is faster than scenario iii with α equal to 8, it is by less than 3% from 10 to 100 supersteps and less than 1% for more than 500 supersteps. It is also important to observe that with 26 processes and 2000 supersteps, scenario iii presents a performance better than scenario i with all α values.

In a second experiment we analyze the recovery after an unexpected exit of a machine from the CON. In this context we evaluate scenario iii with 89 processes running, 2000 supersteps and α is equal to 16, simulating an exit in different supersteps as can be seen in Table 2. An exit on superstep 9 for instance did not cause any gain because there was no migration and consequently no checkpoint. On the other hand, when there is a bigger amount of superstep and a closer checkpoint, like occurred in the last case with an error in the superstep 1999, an economy of more than 57% in time could be obtained.

Table 2: Time in seconds with and without checkpoint according to the supersteps with failure

Superstep with failure	1999	999	499	199	49	9
With checkpoint	29178	15925	6904	1507	813	454
Without checkpoint	68742	40246	15244	3054	1684	454

6. CONCLUSION

This article presented BSPonP2P as an alternative to run BSP applications on PDG. To the best of our knowledge, the proposed model is the first that joins the aforementioned programming model and the collaborative execution environment. Process rescheduling and checkpointing management is the BSPonP2P's scientific contribution. Thanks to both strategies, we demonstrated that the word "efficiency" referred in the problem statement means here performance and fault tolerance. Besides presenting situations in which BSPonP2P outperforms the simple execution of a BSP application, most results using Grid5000 clusters showed an average overhead of 1.09% when using process rescheduling and checkpoiting. We classify this rate as positive to BSPonP2P, because an application must not be restarted from the scratch when any fault occurs. Future research should evaluate BSPonP2P with process replication in order to launch copies of a process at specific superstep to run concurrently, so helping at both performance and fault tolerance areas.

Acknowledgments

The authors would like to thank to the following Brazilian agencies: CAPES, CNPq and FAPERGS.

7. REFERENCES

[1] E. Anceaume, M. Gradinariu, A. Datta, G. Simon, and A. Virgillito. A semantic overlay for self-peer-to-peer publish/subscribe. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 22–22, 2006.

[2] M. Balasubramaniam, N. Sukhija, F. Ciorba, I. Banicescu, and S. Srivastava. Towards the

[2]Details about computing resources and network connections can be found at http://www.grid5000.fr

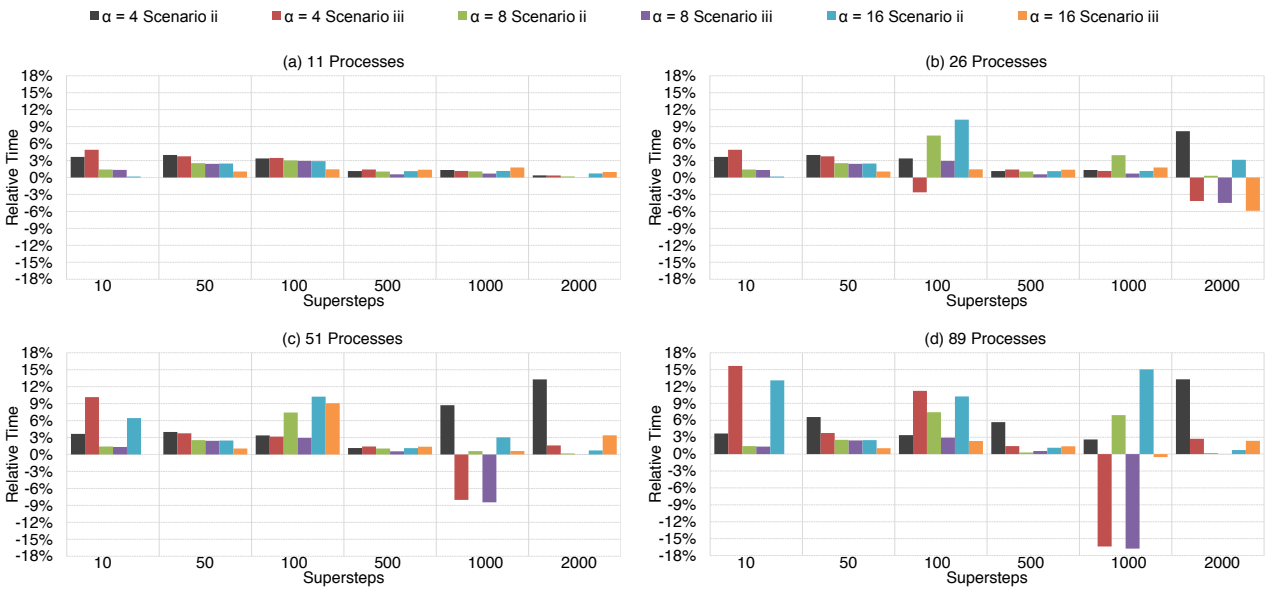

Figure 2: Relative Time change of scenarios ii and iii in comparison with scenario i

scalability of dynamic loop scheduling techniques via discrete event simulation. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 1343–1351, 2012.

[3] R. Camargo, F. Castor, and F. Kon. Reliable management of checkpointing and application data in opportunistic grids. *Journal of the Brazilian Computer Society*, 16(3):177–190, 2010.

[4] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load balancing in dynamic structured p2p systems. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2253–2262 vol.4, March 2004.

[5] B. Hendrickson. Computational science: Emerging opportunities and challenges. *Journal of Physics: Conference Series*, 180(1):012013, 2009.

[6] K. Khan, K. Qureshi, and M. Abd-El-Barr. An efficient grid scheduling strategy for data parallel applications. *The Journal of Supercomputing*, 68(3):1487–1502, 2014.

[7] Z. Khayyat, K. Awara, A. Alonazi, H. Jamjoom, D. Williams, and P. Kalnis. Mizan: a system for dynamic load balancing in large-scale graph processing. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pages 169–182, New York, NY, USA, 2013. ACM.

[8] A. F. Leite, H. C. Mendes, L. Weigang, A. C. M. A. Melo, and A. Boukerche. An architecture for p2p bag-of-tasks execution with multiple task allocation policies in desktop grids. *Cluster Computing*, 15(4):351–361, 2012.

[9] L. Lin, K. Koyanagi, T. Tsuchiya, T. Miyosawa, and H. Hirose. Improving routing load balance on chord. In *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, pages 733–738, Feb 2014.

[10] R. d. R. Righi, L. Graebin, and C. A. da Costa. On the replacement of objects from round-based applications over heterogeneous environments. *Software: Practice and Experience*, pages n/a–n/a, 2014.

[11] C. Schepke and N. Maillard. Performance improvement of the parallel lattice boltzmann method through blocked data distributions. In *Computer Architecture and High Performance Computing, 2007. SBAC-PAD 2007. 19th International Symposium on*, pages 71–78, Oct 2007.

[12] J. Sentís, F. Solsona, D. Castellà, and J. Rius. Discop2p: an efficient p2p computing overlay. *The Journal of Supercomputing*, 68(2):557–573, 2014.

[13] K. Shudo, Y. Tanaka, and S. Sekiguchi. P3: P2p-based middleware enabling transfer and aggregation of computational resources. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 1, pages 259–266 Vol. 1, 2005.

[14] B. H. Son, S. woo Lee, and H.-Y. Youn. Prediction-based dynamic load balancing using agent migration for multi-agent system. In *High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on*, pages 485–490, 2010.

[15] D. Wu, Y. Tian, and K.-W. Ng. On the effectiveness of migration-based load balancing strategies in dht systems. In *Computer Communications and Networks, 2006. ICCCN 2006. Proceedings.15th International Conference on*, pages 405–410, 2006.

[16] H. Zhao, X. Liu, and X. Li. A taxonomy of peer-to-peer desktop grid paradigms. *Cluster Computing*, 14(2):129–144, 2011.